#2

# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE



In re ~~PATENT~~ APPLICATION of
Inventor(s): LIM

| Appln. No.: | 09 | 878,246 |
|---|---|---|
| Series Code | ↑ | ↑ Serial No. |

Group Art Unit: Unassigned

Filed: June 12, 2001

Examiner: Unassigned

Title: ENCRYPTION APPARATUS USING DATA ENCRYPTION STANDARD ALGORITHM

Atty. Dkt. P 281196 | P00HA009/US/JS

M# Client Ref

Date: August 30, 2001

**SUBMISSION OF PRIORITY DOCUMENT IN ACCORDANCE WITH THE REQUIREMENTS OF RULE 55**

Hon. Asst Commissioner of Patents
Washington, D.C. 20231

Sir:

Please accept the enclosed certified copy(ies) of the respective foreign application(s) listed below for which benefit under 35 U.S.C. 119/365 has been previously claimed in the subject application and if not is hereby claimed.

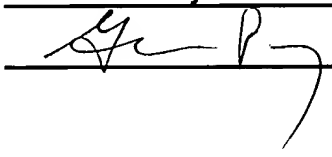
| Application No. | Country of Origin | Filed |
|---|---|---|
| 2000-32173 | Korea | June 12, 2001 |

Respectfully submitted,

Pillsbury Winthrop LLP
**Intellectual Property Group**

1600 Tysons Boulevard
McLean, VA 22102
Tel: (703) 905-2000
Atty/Sec: gjp/dlh

**By Atty:** Glenn J. Perry **Reg. No.** 28458

**Sig:** 

**Fax:** (703) 905-2500
**Tel:** (703) 905-2161

<Priority Document Translation>





THE KOREAN INDUSTRIAL

PROPERTY OFFICE

This is to certify that annexed hereto is a true copy from the records of the Korean Industrial Property Office of the following application as filed.

Application Number : 2000-32173 (Patent)

Date of Application : June 12, 2000

Applicant(s) : HYUNDAI ELECTRONICS INDUSTRIES CO., LTD.

October 30, 2000

COMMISSIONER

HA009



# 대 한 민 국 특 허 청
# KOREAN INDUSTRIAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Industrial Property Office.

| | |
|---|---|
| 출 원 번 호 : Application Number | 특허출원 2000년 제 32173 호 |
| 출 원 년 월 일 : Date of Application | 2000년 06월 12일 |
| 출 원 인 : Applicant(s) | 현대전자산업주식회사 |

2000 년 10 월 30 일

특 허 청
COMMISSIONER

Inventor: LIM
Application No. 09/878,246
Filed: June 12, 2001
Attorney Docket: 281196

【서류명】 특허출원서

【권리구분】 특허

【수신처】 특허청장

【제출일자】 2000.06.12

【발명의 명칭】 데이터 암호화 표준 알고리즘을 이용한 암호화 방법 및 장치

【발명의 영문명칭】 Encryption method and device using data encryption standard algorithm

【출원인】

【명칭】 현대전자산업주식회사

【출원인코드】 1-1998-004569-8

【대리인】

【성명】 박해천

【대리인코드】 9-1998-000223-4

【포괄위임등록번호】 1999-008448-1

【대리인】

【성명】 원석희

【대리인코드】 9-1998-000444-1

【포괄위임등록번호】 1999-008444-1

【발명자】

【성명의 국문표기】 임영원

【성명의 영문표기】 LIM,Young Won

【주민등록번호】 621128-1067119

【우편번호】 467-850

【주소】 경기도 이천시 대월면 현대전자사원아파트 106-1302

【국적】 KR

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
박해천 (인) 대리인
원석희 (인)

【수수료】

【기본출원료】 20 면 29,000 원

【가산출원료】 19 면 19,000 원

| | | | |
|---|---|---|---|
| 【우선권주장료】 | 0 | 건 | 0 원 |
| 【심사청구료】 | 7 | 항 | 333,000 원 |
| 【합계】 | 381,000 원 | | |
| 【첨부서류】 | 1. 요약서·명세서(도면)_1통 | | |

# 【요약서】

【요약】

본 발명은 데이터 암호화 표준 알고리즘을 사용한 암호화 방법 및 암호화 장치에 관한 것으로, DES 연산에 필요한 클럭 수를 8 사이클로 줄였기 때문에 성능면에서 속도가 빠르고 또한 면적과 전력 소모를 최소화하는 암호화 방법 및 장치를 제공하는데 그 목적이 있다. 이를 위하여 본 발명은 데이터 표준 암호화 알고리즘을 이용한 암호화 방법에 있어서, 64비트의 입력 데이터를 8개로 분할하여 순차적으로 8비트씩 입력받아 제1클럭에 따라서 왼쪽 입력 버퍼 레지스터와 오른쪽 입력 버퍼 레지스터에 각각 32비트의 입력데이터를 수집하여 저장하는 제1단계; 상기 왼쪽 입력 버퍼 레지스터와 상기 오른쪽 입력 버퍼 레지스터로부터 각각의 32비트의 데이터 블록을 제1사이퍼 함수와 제2사이퍼 함수에 교대로 입력시켜 8라운드의 암호화 연산을 수행하는 제2단계; 및 상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터와 오른쪽 출력 버퍼 레지스터를 통하여 각각 8비트씩 분배하여 출력하는 제3단계를 포함하여 이루어진다.

【대표도】

도 6

【색인어】

입력 버퍼 레지스터, 출력 버퍼 레지스터, 사이퍼 함수부, 확장 치환부, S-Box 치환부, P-Box 치환부.

【명세서】

【발명의 명칭】

데이터 암호화 표준 알고리즘을 이용한 암호화 방법 및 장치{Encryption method and device using data encryption standard algorithm}

【도면의 간단한 설명】

도1은 DES 알고리즘을 설명하기 위한 블럭도,

도2는 보조키(Subkey)를 발생하는 키 스케쥴(Key Schedule)을 설명하기 위한 블럭도,

도3은 일반적인 DES 코아의 아키텍쳐와 S-Box의 상세한 블럭도,

도4는 3단계 매크로 파이프라인 DES 아키텍쳐와 동작 순서를 나타내는 블럭도,

도5a는 일반적인 DES 코아 아키텍쳐를 나타내는 블럭도,

도5b는 가상의 2단계 마이크로 파이프라인 DES 코아의 아키텍쳐를 나타내는 블럭도,

도6은 본 발명의 매크로 파이프라인과 마이크로 파이프라인 구조를 사용한 DES 아키텍쳐의 블럭도,

도7은 본 발명의 시분할 사이퍼 함수의 상세한 블럭도,

도8은 시분할 사이퍼 함수의 키 스케쥴러(Key Scheduller)의 아키텍쳐를 나타내는 블럭도,

도9는 본 발명의 매크로 파이프라인과 마이크로 파이프라인 구조를 사용한 DES 아키텍쳐의 동작 순서를 나타내는 타이밍도,

도10은 시분할 사이퍼 함수의 키 스케쥴러의 동작 순서를 나타내는 타이밍도,

도11은 동일한 클럭으로 구동되는 3단계 매크로 파이프라인에서 종래기술의 16라운드 DES 아키텍쳐와 본 발명의 8 라운드 DES 아키텍쳐의 성능을 비교한 도면,

도12는 종래기술의 16라운드 DES 아키텍쳐를 입출력 과정에서 구동되는 클럭보다 두 배 빠른 클럭으로 구동하여 전력 소모를 증가시킴으로써 본 발명과 같은 성능을 얻을 수 있음을 나타내는 도면.

* 도면의 주요 부분에 대한 부호의 설명 *

610 : 왼쪽 입력 버퍼 레지스터 620 : 오른쪽 입력 버퍼 레지스터

631, 634 : 사이퍼 함수부 633 : 왼쪽 레지스터

636 : 오른쪽 레지스터 640 : 왼쪽 출력 버퍼 레지스터

650 : 오른쪽 출력 버퍼 레지스터

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<19> 본 발명은 암호화 방법 및 장치에 관한 것으로, 특히 데이터 암호화 표준 알고리즘을 이용한 암호화 방법 및 장치에 관한 것이다.

<20> 일반적으로 데이터 암호화 표준(DES : Data Encryption Standard, 이하 DES라 칭함) 알고리즘은 가장 널리 쓰이고 있는 암호화 방식으로 네트워킹 사용이 증가함에 따라 그 중요성을 더해 가고 있다. 특히, 보안 인터넷 응용이나 원격 접근 서버나 케이블 모뎀과 위성용 모뎀 등의 분야에서 많이 이용되고 있다.

<21> DES는 기본적으로 64비트 블럭의 입력 및 출력을 가지는 64비트 블럭 암호이며, 64비트의 키 블럭 중 56비트가 암호화 및 복호화에 사용되고, 나머지 8비트는 패리티 검사용으로 사용된다. 또한. 64비트의 평문(Plain Text) 블럭과 56비트의 키(Key)를 입력으로 해서 64비트의 암호문(Ciper Text) 블럭을 출력하는 암호화 방식이다.

<22> 도1은 DES 알고리즘을 설명하기 위한 블럭도이다.

<23> 상기 도1을 참조하면, 종래기술의 암호화 장치의 알고리즘은 먼저 64비트의 평문(Plain Text) 블럭을 초기 치환(IP : Initial Permutation)에 의해 치환을 수행하는 초기치환부(110)와, 다음에 상기 치환부(110)의 64비트의 블럭을 두개의 32비트 블럭으로 나누어 왼쪽변수($L_i$)와 오른쪽변수($R_i$)에 저장하고 사이퍼(Cipher) 함수 f로 수행되는 16라운드의 곱 변형(Product Transformation)과 왼쪽변수($L_i$)와 오른쪽변수($R_i$)를 매 라운드마다 교환하여 16라운드의 블록 변형(Block Transformation)을 수행하는 기본연산부(120)와, 16라운드에 걸친 변형이 끝난 후 역초기 치환($IP^{-1}$)을 거쳐서 암호화된 암호문(Cipher Text)을 출력하는 역초기치환부(130)로 구성되어 있다.

<24> 상기 기본연산부(120)에서의 곱 변형은 상기 초기치환부(110)에서 나뉘어진 32비트의 블럭 중에서 오른쪽변수($R_i$)에 저장된 데이터를 키 스케쥴(Key Schedule)에 의해서 생성된 보조키(Subkey) $K_i$와 함께 입력시켜서 암호화 연산을 수행하는 사이퍼 함수부(f)(121)와, 상기 사이퍼 함수 f의 결과를 왼쪽변수(L

$_i$)와 함께 배타적 논리합하는 익스쿠르시브-오아부(122)로 구성되어 있다.

<25> 상기 익스쿠르시브-오아부(122)의 32비트의 데이터는 오른쪽변수($R_{i+1}$)에 저장되고 상기 오른쪽변수($R_i$)에 저장된 32비트의 데이터는 다음 라운드의 왼쪽변수($L_{i+1}$)에 교환(Swapping)되어 저장된다. 이러한 1라운드의 연산이 반복되어 16라운드가 수행되는 것이다.

<26> 초기치환부(110)을 거친 64비트의 평문(Plain Text) 블럭을 둘로 나뉘어 왼쪽레지스터($L_0$)와 오른쪽레지스터($R_0$)에 입력하면 16회의 각 라운드는 다음 수학식1과 수학식2로 표현가능하다.

<27> 【수학식 1】

$$L_i = R_{i-1} \qquad i=1,2,.....16$$

<28> 【수학식 2】

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \qquad i=1,2,.....16$$

<29> 도2는 보조키(Subkey)를 발생하는 키 스케쥴(Key Schedule)을 설명하기 위한 블럭도이다.

<30> 상기 도2를 참조하면, 키 스케쥴은 56비트의 키(Key)를 입력받아서 치환하는 치환선택부(PC1 : Permutation Choice 1)(200)와, 상기 치환선택부(200)에 의해서 치환된 56비트의 블럭을 28비트의 두 블럭으로 나누어서 변수 $C_0$와 $D_0$에 저장하고, 변수 $C_i$와 $D_i$에 (i=0, 1, ..., 15) 저장된 값을 각각 왼쪽으로 한자리 또는 두자리씩 쉬프트하는 제1 및 제2쉬프트부(220, 230)와, 상기 제1 및 제2쉬프트부의 쉬프트된 28비트의 두 블럭을 다음 라운드의 변수 $C_{i+1}$과 $D_{i+1}$에 저장하며, 변수 $C_i$와 $D_i$에(i=1,2,..., 16) 저장된 28

비트의 블럭을 제2치환선택부(PC2 : Permutation Choice 2)(240)로 입력시켜서 48비트의 보조키(Subkey) $K_i$를 출력하여 16라운드의 쉬프트및 치환을 수행한다.

<31> 16라운드 동안 $C_i$와 $D_i$는 28자리 수만큼 쉬프트가 되어 $C_0$와 $C_{16}$ 그리고 $D_0$와 $D_{16}$은 서로 같은 데이터가 된다.

<32> 도3은 일반적인 DES 코아의 아키텍쳐와 S-Box의 상세한 블럭도이다.

<33> 상기 도3을 참조하면, DES 코아는 32비트의 텍스트 블럭을 저장하고 있는 오른쪽레지스터(R(i-1))로부터 32비트의 데이터를 입력받아 48비트의 데이터로 확장 치환하는 확장치환부(310)와, 상기 확장치환부의 48비트의 데이터를 입력받고 키 스케쥴(Key Schedule)로부터의 보조키($K_i$)를 입력받아 배타적 논리합 연산을 수행하는 익스쿠르시브-오아부(320)와, 상기 익스쿠르시브-오아부(320)로부터의 48비트의 데이터를 32비트의 데이터로 대치 치환하는 S-Box 치환부(330)와, 상기 S-Box 치환부(330)의 32비트의 데이터를 복사 치환하는 P-Box 치환부(340)와, 상기 P-Box 치환부의 32비트의 데이터와 왼쪽레지스터(L(i-1))에 저장되어 있는 32비트의 데이터를 입력받아 배타적 논리합하는 익스쿠르시브-오아부(350)를 구비한다.

<34> 키 스케쥴(Key Schedule)은 치환선택부(PC1 : Permutation Choice 1)(360)으로부터 56비트의 키(Key)를 입력받아서 28비트의 두 블럭으로 나누어서 각각 왼쪽으로 한자리 또는 두자리씩 쉬프트하는 쉬프트부(370, 380)와, 상기 28비트의 두 블럭을 입력시켜 48비트의 보조키를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(390)를 구비한다.

<35> 구체적으로, 상기 S-Box 치환부는 48비트의 입력을 받아서 32비트의 출력을 생성하

는 8 개의 S-Box로 구성되어 있다. 즉, 48비트의 데이터는 8 개의 6비트 데이터로 분할되어 8 개의 S-Box에 입력된다. 이 8 개의 S-Box들은 8 개의 출력을 내보냄으로써 48 비트를 32비트로 줄인다. S-Box 치환부(330)는 테이블 룩-업(Look-up) 방식으로 대치됨으로써 프로그램가능 논리 어레이(PLA)나 롬(ROM)과 같은 기억장치를 필요로 한다. 6비트의 입력에 대하여 4비트를 출력하기 때문에 각 S-Box는 64 ×4 의 기억 용량이 필요하며 전체적으로 8개의 S-Box로 구성되어 있으므로 8 ×64 ×4의 기억장치가 필요하다. 따라서 전체적으로 칩에서 차지하는 면적이 상대적으로 크다

<36> 일반적으로 주어진 키에 대하여 암호화 해야 될 데이터가 다수인 경우가 많다. 이때 파이프라인을 사용하여 암호화하는 성능을 높일 수 있다. DES 아키텍쳐에서 사용되는 파이프라인은 적용되는 레벨에 따라 마이크로(Micro) 파이프라인과 매크로(Macro) 파이프라인의 두 종류로 구분할 수 있다.

<37> 먼저, 마이크로 파이프라인은 DES 코아 자체의 반복적인 16 라운드의 연산을 파이프라인화 하는 것으로 최대 16단계까지 가능하다. M 단계 파이프라인을 사용하면 M 개의 비암호화 데이터를 동시에 암호화 할 수 있어 처리능력비가 M 배 증가한다. 그러나 이때 M 개의 기본 연산부가 동시에 수행되기 때문에 일어나는 데이터 컨텐션(Contention) 문제를 피하기 위해서 M 개의 S-Box 치환부가 필요하다. 따라서 S-Box 치환부의 추가구현으로 인한 면적의 증가와 M 값에 상관없이 M 단계 파이프라인은 항상 16 클럭 사이클의 레이턴시(Latency)를 가진다는 문제점이 발생한다.

<38> 도4는 3단계 매크로 파이프라인 DES 아키텍쳐와 동작 순서를 나타내는 블럭도이다.

<39> 상기 도4를 참조하면, 3단계 매크로 파이프라인 DES 아키텍쳐는 선입선출(FIFO)의 방식으로 입력되는 64비트의 데이터를 각각 8 개의 입력 버퍼 레지스터(Input Buffer

Register : IBR, 이하 IBR이라 칭함)에 순차적으로 저장하는제1단계와, 상기 IBR에 저장되어 있는 64비트의 평문 블럭을 입력받아 DES 코아를 통하여 암호화 연산을 수행하여 64 비트의 암호문을 출력하는 제2단계와, 상기 DES 코아로부터 64비트의 암호문을 입력받아 선입선출(FIFO)의 방식으로 출력하는 8 개의 출력 버퍼 레지스터(Output Buffer Register : OBR, 이하 OBR이라 칭함)에 저장하는 제3단계로 이루어진다.

<40> 매크로 파이프라인의 주기는 입력 및 출력 과정에 걸리는 시간과 DES 연산시간 중 최대값으로 결정되며, 이들 시간이 동일할 때 최대의 효과를 얻을 수 있다.

<41> DES 코아에 입력되는 데이터의 속도는 DES 코아보다도 DES 코아 외부의 전체 시스템 측면에서 결정된다. 네트워크에 사용되는 DES의 경우 변조기 및 복조기의 최대 전송 속도 및 외부 호스트 마이크로 프로세서의 속도 등을 고려해서 정해진다.일반적으로 DES에 입출력되는 속도는 느리다. 그리고 DES 코아 외부의 시스템에서 데이터는 대부분 바이트(8비트) 단위로 이동되며 DES 코아는 64비트의 입력 데이터를 64비트로 암호화하여 출력하기 때문에, 8 개의 입력 바이트들을 모아서 DES 코아에 전달하는 입력 버퍼 레지스터(IBR)와 64비트의 DES 출력을 바이트 단위로 전달하기 위한 출력 버퍼 레지스터(OBR)를 사용하여 8 클럭 사이클 동안 데이터 포맷팅을 수행하여야 한다.

<42> 이상에서 살펴본 종래의 마이크로 파이프라인 구조와 매크로 파이프라인 구조는 면적이 크고, DES에 입출력되는 속도가 느리다는 문제점이 있다.

【발명이 이루고자 하는 기술적 과제】

<43> 본 발명은 상기와 같은 종래기술의 문제점을 해결하기 위하여 안출된 것으로써,

DES 연산에 필요한 클럭 수를 8 사이클로 줄였기 때문데 성능면에서 속도가 빠르고 또한 면적과 전력 소모를 최소화하는 암호화 방법 및 장치를 제공하는데 그 목적이 있다.

【발명의 구성 및 작용】

<44> 상기 목적을 달성하기 위하여 본 발명의 암호화 방법은 데이터 표준 암호화 알고리즘을 이용한 암호화 방법에 있어서, 64비트의 입력 데이터를 8개로 분할하여 순차적으로 8비트씩 입력받아 제1클럭에 따라서 왼쪽 입력 버퍼 레지스터와 오른쪽 입력 버퍼 레지스터에 각각 32비트의 입력데이터를 수집하여 저장하는 제1단계; 상기 왼쪽 입력 버퍼 레지스터와 상기 오른쪽 입력 버퍼 레지스터로부터 각각의 32비트의 데이터 블록을 제1 사이퍼 함수와 제2사이퍼 함수에 교대로 입력시켜 8라운드의 암호화 연산을 수행하는 제2단계; 및 상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터와 오른쪽 출력 버퍼 레지스터를 통하여 각각 8비트씩 분배하여 출력하는 제3단계를 포함하여 이루어진다.

<45> 이하, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 본 발명의 기술적 사상을 용이하게 실시할 수 있을 정도로 상세히 설명하기 위하여, 본 발명의 가장 바람직한 실시예를 첨부한 도면을 참조하여 설명하기로 한다.

<46> 도5a는 일반적인 DES 코아 아키텍처를 나타내는 블럭도이고, 도5b는 가상의 2단계 마이크로 파이프라인 DES 코아의 아키텍처를 나타내는 블럭도이다.

<47> 상기 도5a를 참조하면, 일반적인 DES 코아에서는 초기 치환을 거친 32비트의 평문 블럭 두 개인 $L_0$와 $R_0$가 서로 같은 클럭에 의해 레지스터에 저장되어 $R_0$를 사이퍼 함수

연산을 하고 이를 $L_0$와 배타적 논리합 연산을 하는 곱 변형과 각 라운드에서 왼쪽 레지스터와 오른쪽 레지스터를 교환(swapping)하여 저장하는 블록 변형을 수행하게 되는데 해당 레지스터의 입력 과정 때문에 항상 16 사이클이 걸린다는 문제점이 있다. 상기 수학식1을 상기 수학식2에 대입하여 얻은 식을 수학식3으로 나타내면 다음과 같다.

<48> 【수학식 3】

$$R_i = R_{i-2} \oplus f(R_{i-1}, K_i) \quad i=1, 2, \ldots, 16$$

<49> 상기 도5b를 참조하면, 2단계 마이크로 파이프라인 DES 코아는 32비트의 초기 데이터($R_0$)와 키 스케쥴(Key Schedule)에서 생성된 보조키($K_A$)를 입력받아 사이퍼 함수($f_A$)(550)를 통하여 암호화 연산을 수행하고 이를 초기 데이터 $L_0$와 배타적 논리합 연산을 수행하며 상기 연산 결과($R_1$)를 오른쪽레지스터(B0)(510)에 저장하는 제1단계와, 오른쪽레지스터에 저장된 32비트의 데이터($R_1$)와 키 스케쥴(Key Schedule)에서 생성된 보조키($K_B$)를 입력받아 사이퍼 함수($f_B$)(520)를 통하여 암호화 연산을 수행하고 이를 초기 데이터 $R_0$와 배타적 논리합 연산을 수행하며 상기 연산 결과($R_2$)를 왼쪽 레지스터(A0)(540)에 저장하는 제2단계로 이루어진다.

<50> 2단계 마이크로 파이프라인 DES 코아는 상기 수학식3을 사용하여 암호화 연산을 수행하며, 서로 반전된 클럭을 사용하여 데이터를 저장하기 때문에 한 레지스터가 저장된 값을 유지하는 한 주기 중간에 다른 레지스터가 새로운 값을 저장한다. 따라서, 한 주기 동안 한 레지스터는 한 개의 값을 유지하며 다른 레지스터에서는 서로 다른 값을 반주기씩 엑세스가 가능하다. 순열 $R_i$ (i=1,2,...,16) 값들을 두 레지스터에 교대로 저장하면 모든 인접한 $R_i$값들을 반주기 동안 엑세스가 가능하다. 상기 수학식3에서 새로이 계산되

는 $R_i$값들을 교대로 두 레지스터에 저장하려면, 이전에 계산된 $R_{i-1}$과 $R_{i-2}$ 값들을 두 레지스터에서 반 주기 동안 엑세스가 가능하기 때문에 이 반주기 동안에 사이퍼함수 연산이 수행되면 된다. 즉 한 주기 동안 두 개의 사이퍼 함수 연산이 반 주기씩 시분할 되며, 두 개의 시분할된 사이퍼 함수는 한 개의 S-Box 치환부만을 사용하여 구현 가능하다.

<51> 즉, 상기 도5b는 반전된 클럭을 사용하는 두개의 레지스터와 시분할된 사이퍼 함수 연산으로 구성된 마이크로 파이프라인이다. 이 마이크로 파이프라인은 두 개의 비암호화 입력 데이터를 동시에 암호화하지 않지만, 한 개의 비암호화 입력 데이터를 암호화할 때 소요되는 클럭 수를 줄여서 전력 소모를 최소화하는 효과가 있는 가상의 2단계 마이크로 파이프라인이다. 그리고, 한 개의 S-Box 치환부로 구현 가능한 시분할 사이퍼 함수의 연산으로 면적을 최소화하였기 때문에, 이 마이크로 파이프라인은 집적화시 도5a의 아키텍쳐와 거의 동일한 크기를 가진다.

<52> 도6은 본 발명의 매크로 파이프라인과 마이크로 파이프라인 구조를 사용한 DES 아키텍쳐의 블럭도이다.

<53> 상기 도6을 참조하면, 본 발명의 DES 아키텍쳐는 64비트의 입력 데이터를 8개로 분할하여 순차적으로 8비트씩 입력받아 제1클럭(CLK1)에 따라서 왼쪽 입력 버퍼 레지스터(IBR(L))(610)와 오른쪽 입력 버퍼 레지스터(IBR(R))(620)에 각각 32비

트의 입력데이터를 수집하여 저장하는 제1단계와, 상기 왼쪽 입력 버퍼 레지스터와 상기 오른쪽 입력 버퍼 레지스터로부터 각각의 32비트의 데이터 블록을 제1사이퍼 함수와 제2사이퍼 함수에 교대로 입력시켜 8라운드의 암호화 연산을 수행하는 제2단계와, 상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터(OBR(L))(640)과 오른쪽 출력 버퍼 레지스터(OBR(R))(650)을 통하여 각각 8비트씩 분배하여 출력하는 제3단계로 이루어진다.

<54> 구체적으로, 상기 제2단계는 왼쪽 입력 버퍼 레지스터(IBR(L))(610)에 저장되어 있던 32비트의 데이터를 $A_i$로 하고 키 스케쥴로부터 생성된 보조키($K_A$)를 입력받는 사이퍼 함수부($f_A$)(634)를 통하여 암호화 연산을 하고 연산 결과를 상기 오른쪽 입력 버퍼 레지스터(IBR(R))(620)에 저장되어 있던 32비트의 데이터를 입력으로 한 $B_i$와 배타적 논리합 연산(635)을 수행하여 연산 결과를 제2클럭(~CLK1)에 의해서 오른쪽 레지스터(B0)(636)에 저장하는 단계와, 오른쪽 입력 버퍼 레지스터(IBR(R))(620)에 저장되어 있던 32비트의 데이터를 입력으로 한 $B_i$와 키 스케쥴로부터 생성된 보조키($K_B$)를 입력받는 사이퍼 함수부($f_B$)(631)를 통하여 암호화 연산을 하고 연산 결과를 상기 왼쪽 입력 버퍼 레지스터(IBR(L))(610)에 저장되어 있던 32비트의 데이터를 입력으로 한 $A_i$와 배타적 논리합 연산(632)을 수행하여 연산 결과를 제1클럭(CLK1)에 의해서 왼쪽 레지스터(A0)(633)에 저장하는 단계로 이루어진다.

<55> DES 코아의 입력과 출력은 64비트인 반면, DES 외부 시스템에서 데이터는 일반적으로 8비트씩 입출력되기 때문에, 8 개의 8비트를 수집하는 입력 과정(제1단계)에 64비트의 입력 버퍼 레지스터 IBR과 64비트의 출력을 8비트씩 분배하는 출력 과정(제3단계)에 64비트의 출력 버퍼 레지스터 OBR을 사용하여 입력 과정, DES 연산 과정(제2단계), 출력

과정의 3단계 매크로 파이프라인으로 저속의 입출력 과정의 레이턴시(Latency)를 숨길 수 있다. 또한, 도5a와 같은 일반적인 DES 코아 아키텍처에서는 $R_i$와 $L_i$ 값을 16번 계산하여 두 레지스터에 각각 저장하여 암호화를 수행하는 반면, 도6에 있는 본 발명의 아키텍처에서는 16개의 $R_i$ 값만을 계산하여 교대로 레지스터에 저장하기 때문에 각 레지스터는 8 개의 $R_i$ 값만을 저장하여 8 라운드만으로 암호화를 수행할 수 있다. 레지스터 A0의 입력은 IBR(L) ⊕ $f_B$이거나 A0 ⊕ $f_B$의 연산 결과이며, 레지스터 A0의 출력은 A0 ⊕ $f_B$의 연산에 사용되거나 레지스터 OBR(L)에 저장된다. 마찬가지로, 레지스터 B0의 입력은 IBR(R) ⊕ $f_A$이거나 B0 ⊕ $f_A$의 연산 결과이며, 레지스터 B0의 출력은 B0 ⊕ $f_A$의 연산에 사용되거나 레지스터 OBR(R)에 저장된다.

<56> 도7은 본 발명의 시분할 사이퍼 함수의 상세한 블럭도이다.

<57> 상기 도7을 참조하면, 시분할 사이퍼 함수는 32비트의 데이터 블럭 $A_i$를 입력받아 48비트의 데이터로 확장하는 제1확장치환부(710)와, 상기 제1확장치환부(710)의 48비트의 데이터와 키 스케쥴로부터 생성된 보조키($K_A$)를 입력받는 제1배타적 논리합 연산부(730)와, 32비트의 데이터 블럭 $B_i$를 입력받아 48비트의 데이터로 확장하는 제2확장치환부(720)와, 상기 제2확장치환부(720)의 48비트의 데이터와 키 스케쥴로부터 생성된 보조키($K_B$)를 입력받는 제2배타적 논리합 연산부(740)와, 상기 제1 및 제2 배타적 논리합 연산부(730, 740)로부터 48비트의 데이터를 입력받아 선택신호(Select)에 의해 그 중 하나를 선택하는 멀티플렉서(750)와, 상기 멀티플렉서(750)로부터 48비트의 데이터를 입력받아 저장하고 32비트의 데이터로 대치하는 S-Box 치환부(760)와, 상기 S-Box 치환부(760)로부터의 32비트의 데이터를 복사 치환하는 P-Box 치환부(770)와 상기 P-Box 치환부(770)로부터의 32비트의 데이터를 선택신호(Select)에 의해 두 개의 32비트의 데이터($f_A$

, $f_B$)로 분배하는 디멀티플렉서(780)를 구비한다.

<58> 선택신호(Select)에 의해서 동작되는 멀티플렉서(750)와 디멀티플렉서(780)는 상기 제1클럭(CLK1) 주기의 전반부에서는 32비트의 데이터 $f_A$를 연산하여 출력하게 하며, 상기 제1클럭(CLK1) 주기의 후반부에서는 $f_B$를 연산하여 출력하게 한다. 즉, 시분할 사이퍼 함수는 제1클럭(CLK1) 주기의 전반부에서 32비트의 데이터 $A_i$와 보조키 $K_A$를 입력해서 확장 치환부(710), 배타적 논리합 연산부(730), S-Box 치환부(760), 그리고 P-Box 치환부(770)로 구성되는 사이퍼 함수를 연산하여 $f_A$로 출력하고, 마찬가지로 상기 제1클럭(CLK1) 주기의 후반부에서 32비트 데이터 $B_i$와 보조키 $K_B$를 입력받아 사이퍼 함수 연산 결과를 32비트의 데이터 $f_B$로 출력한다. 확장 치환부(710, 720)와 P-Box 치환부(770)는 와이어링으로 구현가능하지만 S-Box 치환부(760)는 롬(ROM)이나 프로그램 가능 논리 어레이(PLA)로 구현한다. 한개의 S-Box 치환부를 사용해서 한 주기 동안 두 개의 사이퍼 함수를 연산하는 본 발명은 면적을 최소화시킨다.

<59> 도8은 시분할 사이퍼 함수의 키 스케쥴러(Key Scheduller)의 아키텍처를 나타내는 블럭도이다.

<60> 상기 도8을 참조하면, 키 스케쥴러는 56비트의 키(Key)를 입력받아서 치환하는 제1치환선택부(PC1 : Permutation Choice 1)(800)와, 상기 제1치환선택부(800)에 의해서 치환된 56비트의 블럭을 28비트의 두 블럭으로 나누어서 제1클럭(CLK1)에 의해서 저장하는 제1레지스터($C_A$)(810) 및 제2레지스터($D_A$)(820)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키(Key) 블럭을 각각 왼쪽으로 두자리, 세자리 또는 네자리씩 쉬프트하는 제1쉬프트부(830) 및 제2쉬프트부(840)와, 상기 제1레지스터 및 제2레지스터의 28비트의 키

(Key) 블럭을 입력받아 48비트의 보조키($K_A$)를 생성하는 제2치환선택부(PC2 : Permutation Choice 2)(850)로 8라운드로 동작하는 제1유닛이 구성되고 제1유닛과 똑같은 구성을 갖고 다만 제2클럭(~CLK1)이 입력되는 제2유닛이 더 포함된다.

<61> 기본적인 8 라운드 중에서 각 라운드마다 제1쉬프트부(830) 및 제2쉬프트부(840)가 쉬프트하는 비트 수가 도8의 도표에 도시되어 있다.

<62> 제1 및 제2치환선택부(PC1, PC2)(800, 850)는 와이어링으로 구현 가능하고 쉬프터와 레지스터를 두 개를 사용한다. 하지만 집적화 시 증가된 총 56비트의 레지스터는 S-Box 치환부의 면적에 비해 아주 작다.

<63> 도9는 본 발명의 매크로 파이프라인과 마이크로 파이프라인 구조를 사용한 DES 아키텍쳐의 동작 순서를 나타내는 타이밍도이다.

<64> 상기 도9를 참조하면, 본 발명의 DES 아키텍쳐가 초기 치환을 거친 비암호화 입력 데이터 $(y_0, z_0)$, $(a_0, b_0)$, $(c_0, d_0)$를 순서대로 입력받아 $z_i$, $b_i$, $d_i$, $(i=1,2,\ldots,16)$를 계산하여 $(z_{16}, z_{15})$, $(b_{16}, b_{15})$, $(d_{16}, d_{15})$을 차례로 출력하는 과정 중 비암호화 데이터 $a_0$와 $b_0$로부터 $b_1$, $b_2$, ..., $b_{16}$을 계산하여 $b_{16}$과 $b_{15}$를 출력하는 과정을 중점으로 나타내었다. $a_0$와 $b_0$를 초기 치환 IP를 거친 64비트의 비암호화 블럭이 32비트의 두블럭으로 나뉘어진 것을 나타낸다고 하자. 즉 $a_0=L_0=R_{-1}$이고 $b_0=R_0$이다. 그리고 DES 코아가 계산한 값을 $b_1$, $b_2$, ..., $b_{16}$($b_i=R_i$)라고 한다. $b_i$를 계산하기 전에 미리 키 스케쥴러가 적절한 보조키 $K_i$를 사이퍼 함수에 입력해주도록 하면, 본 발명의 아키텍쳐에서 DES가 연산되는 과정은 다음과 같다.

<65> 먼저, 입력 과정은 시간 $t_0$이전에 8 사이클 동안 바이트 단위로 입력된 데이터를

레지스터 IBR에 수집한다. 그리고 시간 [$t_0$-$t_2$] 구간에서 레지스터 IBR(L)은 $b_0$를, 레지스터 IBR(R)은 $a_0$를 유지한다. 동시에 다음 비암호화 입력 데이터 $c_0$와 $d_0$를 한 바이트씩 레지스터 IBR에 수집한다. 따라서 8 사이클 후인 시간 [$t_{16}$-$t_{18}$]에서 레지스터 IBR은 $c_0$와 $d_0$의 값을 유지한다.

<66> 출력 과정을 살펴보면, 레지스터 OBR은 시간 $t_1$에서 레지스터 AO와 BO에서 $z_{16}$과 $z_{15}$의 값을 로드한 후 역치환된 데이터를 시간 $t_2$에서 한 바이트씩 8 사이클 동안 출력한다. $z_{16}$과 $z_{15}$의 값은 레지스터 OBR에서 시간 [$t_1$-$t_{17}$]구간에서 유지되며, 시간 $t_{17}$에 다시 레지스터 AO와 BO의 값($b_{16}$와 $b_{15}$)을 로드한 후 8 사이클 동안 유지하여 시간 $t_{18}$에서부터 다시 역치환된 출력 데이터를 한 바이트씩 8 사이클 동안 출력한다.

<67> DES 연산과정은 시간 [$t_0$-$t_1$] 구간에서 레지스터 IBR에 저장된 값 $a_0$와 $b_0$를 엑세스 가능하고 시간 [$t_0$-$t_1$]구간에서 보조키 $K_1$을 키 스케쥴러의 출력 $K_A$로부터 입력받으면 시간 [$t_0$-$t_1$]구간에서 사이퍼 함수 $f_A$를 연산하여 시간 $t_1$에서 식 $b_1 = a_0 \oplus f(b_0, K_1)$에 의해 연산된 $b_1$값을 레지스터 BO에 저장할 수 있다.

<68> 또한, 시간 [$t_1$-$t_2$] 구간에서 레지스터 IBR(L)로부터 $b_0$를, 레지스터 BO로부터 $b_1$을 엑세스할 수 있으므로, 시간 [$t_1$-$t_2$]구간에서 보조키 $K_2$를 키 스케쥴러의 출력 $K_B$로부터 입력받으면 시간 [$t_1$-$t_2$]구간에서 사이퍼 함수 $f_B$를 연산하여 시간 $t_2$에서 식 $b_2 = b_0 \oplus f(b_1, K_2)$에 의해 연산된 $b_2$값을 레지스터 AO에 저장할 수 있다.

<69> 또한, 시간 [$t_2$-$t_3$]구간에서 레지스터 AO로부터 $b_2$를, 레지스터 BO로부터 $b_1$을 엑세스할 수 있으므로, 시간 [$t_2$-$t_3$]구간에서 보조키 $K_3$를 키 스케쥴러의 출력 $K_B$로부터 입력받으면 시간 [$t_2$-$t_3$]구간에서 사이퍼 함수 $f_A$를 연산하여 시간 $t_3$에서 식 $b_3 = b_1 \oplus$

$f(b_2, K_3)$에 의해 연산된 $b_3$값을 레지스터 B0에 저장할 수 있다.

<70> 이와 같이 시간 $t_0$로부터 $b_1$값을 계산하기 시작하여 $b_2, b_3, \ldots, b_{15}$를 계산하여 해당 레지스터에 저장하여, 8 사이클 뒤인 시간 $t_{16}$에 $b_{16}$을 레지스터 A0에 저장함으로써 $a_0$와 $b_0$를 입력으로 한 DES 연산을 끝낸다. 동시에 시간 $t_{16}$에서 다시 $c_0$와 $d_0$를 입력으로 한 DES 연산을 수행한다.

<71> 보조키 $K_i$ (i=1,2,...,16)는 치환 선택부 PC1을 거친 56비트의 초기키를 왼쪽과 오른쪽의 28비트로 나누어 i 값에 따라 1, 2, 4, 6, 8, 10, 12, 14, 15, 17, 19, 21, 23, 25, 27, 28(=0) 비트 수 만큼 왼쪽 쉬프트시킨 후 제2치환선택부 PC2에 의해 치환된 48비트 결과이다. 본 발명에서는 사이퍼 함수가 시분할 되어 연산되기 때문에 한 주기 동안 두 개의 사이퍼 함수 연산을 위하여 보조키를 한 주기 동안 두 개 생성하여야 한다. 이를 위하여, 도8과 같이 8 라운드로 동작하는 기본 연산부 두 개를 사용한다. 기본연산부 유닛1은 제1클럭(CLK1)으로 구동되어 보조키 $K_1$, $K_3$, $K_5$, $K_7$, $K_9$, $K_{11}$, $K_{13}$, $K_{15}$를 8 라운드 동안 생성하고, 기본연산부 유닛2는 제2클럭(~CLK1)으로 구동되어 $K_2$, $K_4$, $K_6$, $K_8$, $K_{10}$, $K_{12}$, $K_{14}$, $K_{16}$를 8 라운드 동안 생성한다.

<72> 도10은 시분할 사이퍼 함수의 키 스케쥴러의 동작 순서를 나타내는 타이밍도이다.

<73> 상기 도10을 참조하면, $K_A$와 $K_B$는 시분할 사이퍼 함수의 연산에 필요한 보조키와 엑세스되는 시간 구간을 나타낸다. $TS_A$ 및 $TS_B$는 필요한 보조키를 얻기 위해 PC1을 거친 초기 키가 왼쪽 쉬프터에 의해 쉬프트된 총 비트 수를 나타낸다. 도 10에서 ($C_A$, $D_A$) 및 ($C_B$, $D_B$)는 이 때 레지스터 $C_A$와 $D_A$ 및 $C_B$와 $D_B$의 출력을 치환 선택부 PC2로 치환함으로써 얻을 수 있는 해당 보조키를 나타낸다. $S_A$ 및 $S_B$는 $TS_A$ 및 $TS_B$에 표시된 총 쉬프트된

비트 수를 얻기 위해서 각 라운드 ($P_i$, $Q_i$)에서 쉬프트하는 비트 수를 나타낸다. 본 발명에서 보조키를 생성하는 과정은 다음과 같다.

<74> 먼저, 제 1 라운드 ($P_0$, $Q_0$)에서 $TS_A$ 및 $TS_B$는 1과 2로서 해당 레지스터는 PC1을 거친 초기키를 1비트와 2비트씩 왼쪽 쉬프트된 결과를 출력하여 PC2를 거치면 보조키 $K_1$과 $K_2$를 생성할 수 있다.

<75> 제 2 라운드 ($P_1$, $Q_1$)에서 보조키 $K_3$와 $K_4$를 생성하기 위해서, 즉 $TS_A$=4와 $TS_B$=6을 위해서 제 1 라운드에서 왼쪽 쉬프터는 해당 레지스터를 3(=4-1)비트와 4(=6-2)비트 씩 왼쪽 쉬프트한다.

<76> 제3라운드 ($P_2$, $Q_2$)에서 보조키 $K_5$와 $K_6$를 생성하기 위해서, 즉 $TS_A$=8와 $TS_B$=10을 위해서 제 2 라운드에서 왼쪽 쉬프터는 해당 레지스터를 4(=8-4)비트와 4(=10-6)비트 씩 왼쪽 쉬프트한다.

<77> 이와 같이 각 라운드 ($P_i$, $Q_i$)에서 해당 레지스터를 $S_A$ 또는 $S_B$비트씩 왼쪽 쉬프트하여 제 8 라운드 ($P_7$, $Q_7$)에서 $TS_A$=27와 $TS_B$=28(=0)비트만큼 초기키가 쉬프트된다. 다음에 다시 제 1 라운드로 돌아가기 위해서, 즉 $TS_A$=1과 $TS_B$=2가 되기 위해서, $S_A$=2 및 $S_B$=2가 되어야 한다. 단 DES 연산 초기나 시스템 리셋 후 초기키를 레지스터 $C_A$ 및 $D_A$에 저장할 경우 $S_A$=1이다.

<78> 도11은 종래기술의 16라운드 DES 아키텍처와 본 발명의 8라운드 DES 아키텍처의 성능을 비교한 도면이다.

<79> 상기 도11을 참조하면, 성능면에서 본 발명이 종래 기술보다 두 배 빨리 암호화할 수 있음을 알 수 있다.

<80> 도12는 종래기술이 16라운드 DES 아키텍쳐를 입출력 과정에서 구동되는 클럭보다 두 배 빠른 클럭으로 구동하여 전력 소모를 증가시킴으로써 본 발명과 같은 성능을 얻을 수 있음을 보여주는 도면이다.

<81> 상기 도12를 참조하면, 8라운드로 동작하는 본 발명이 스위칭에 의한 전력 소모를 줄일 수 있음을 알 수 있다.

<82> 본 발명의 아키텍쳐는 다음과 같이 확장 가능하다. 도6의 가상의 2단계 마이크로 파이프라인 아키텍쳐를 N 개 직렬로 연결하여 가상의 2N 단계 마이크로 파이프라인을 구현할 수 있다. 이 확장된 아키텍쳐는 시분할이 되지 않는 사이퍼 함수가 N개 존재하여 N개의 S-Box 치환부를 구현해야 하지만, 동시에 N개의 비암호화 입력 데이터를 8 라운드에 암호화 할 수 있다. 즉 처리능력비가 N 배 증가한다.

<83> 도5b의 아키텍쳐도 M개 직렬로 연결하여 M단계 마이크로 파이프라인을 구현할 수 있다. 이 확장된 아키텍쳐는 M 개의 S-Box 치환부를 구현해야 하지만, 동시에 M 개의 비암호화 입력 데이터를 16라운드에 암호화할 수 있다. 즉 처리능력비가 M배 증가한다. 네 개의 M과 N값에 따른 아키텍쳐를 비교하면 다음 표1과 같다.

<84> 【표 1】

| | 레이턴시 | 처리능력비 | S-Box | 코아 |
|---|---|---|---|---|
| 본 발명의 가상의 2단계 파이프라인 | 8 | 1 | 1개 | 2개 |
| 일반적인 1단계 파이프라인 | 16 | 1 | 1개 | 2개 |
| 본 발명의 가상의 4단계 파이프라인 | 8 | 2 | 2개 | 4개 |
| 일반적인 2단계 파이프라인 | 16 | 2 | 2개 | 4개 |
| 본 발명의 가상의 8단계 파이프라인 | 8 | 4 | 4개 | 8개 |
| 일반적인 4단계 파이프라인 | 16 | 4 | 4개 | 8개 |
| 본 발명의 가상의 16단계 파이프라인 | 8 | 8 | 8개 | 16개 |
| 일반적인 8단계 파이프라인 | 16 | 8 | 8개 | 16개 |

<85> 본 발명의 기술 사상은 상기 바람직한 실시예에 따라 구체적으로 기술되었으나 상기한 실시예는 그 설명을 위한 것이며 그 제한을 위한 것이 아님을 주의하여야 한다. 또한, 본 발명의 기술 분야의 통상의 전문가라면 본 발명의 기술 사상의 범위내에서 다양한 실시예가 가능함을 이해할 수 있을 것이다.

【발명의 효과】

<86> 상기와 같이 본 발명은 16개의 레지스터 값을 계산하여 교대로 해당 레지스터에 저장하기 때문에 16라운드의 DES 연산을 가상의 2 단계 파이프라인 구조를 사용하여 8 라운드로 감소시킨다. 따라서 스위칭에 의한 전력 소모를 줄일 수 있다.또한, 입력과정, DES 연산과정, 출력과정의 3단계 매크로 파이프라인 구조를 사용하여 저속의 입출력 과정에서 발생하는 레이턴시를 DES 연산에 숨기는 매크로 파이프라인의 효율을 극대화시킨다. 또한, 두 개의 사이퍼 함수 연산을 시분할시켜서 S-Box를 한개만 사용함으로써 집적화 시 면적을 최소화시킨다. 또한, 파이프라인 구조를 확장시켜 한번에 처리할 수 있는 비암호화 입력 데이터의 수를 증가시킨다. 또한, 본 발명은 종래 기술의 16라운드 DES 아키텍쳐에 사용되는 클럭을 입출력 과정에서 사용하는 클럭보다 두 배 빠른 클럭을 사용하는 경우에 비해서 전력 소모를 줄일 수 있다. 따라서 저속의 클럭과 최소의 면적 그리고 저전력으로 특징 지워지는 범용 시스템에서 본 발명은 성능을 극대화하면서도 면적을 최소화시키며 전력 소모를 줄인다.

【특허청구범위】

【청구항 1】

데이터 표준 암호화 알고리즘을 이용한 암호화 방법에 있어서,

64 비트의 입력 데이터를 8개로 분할하여 순차적으로 8비트씩 입력받아 제1클럭에 따라서 왼쪽 입력 버퍼 레지스터와 오른쪽 입력 버퍼 레지스터에 각각 32비트의 입력데이터를 수집하여 저장하는 제1단계;

상기 왼쪽 입력 버퍼 레지스터와 상기 오른쪽 입력 버퍼 레지스터로부터 각각의 32비트의 데이터 블록을 제1사이퍼 함수와 제2사이퍼 함수에 교대로 입력시켜 8라운드의 암호화 연산을 수행하는 제2단계; 및

상기 2단계로부터의 32비트의 데이터를 왼쪽 출력 버퍼 레지스터와 오른쪽 출력 버퍼 레지스터를 통하여 각각 8비트씩 분배하여 출력하는 제3단계

를 포함하여 이루어진 암호화 방법.

【청구항 2】

제 1 항에 있어서,

상기 제2단계는,

왼쪽 입력 버퍼 레지스터에 저장되어 있던 32비트의 데이터와 키 스케쥴로부터 생성된 보조키를 입력받는 사이퍼 함수부를 통하여 암호화 연산을 하고 연산 결과를 상기 오른쪽 입력 버퍼 레지스터에 저장되어 있던 32비트의 데이터와 배타적 논리합 연산을 수행하여 연산 결과를 제2클럭에 의해서 오른쪽 레지스터에 저장하는 단계; 및

오른쪽 입력 버퍼 레지스터에 저장되어 있던 32비트의 데이터와 키 스케쥴로부터 생성된 보조키를 입력받는 사이퍼 함수부를 통하여 암호화 연산을 하고 연산 결과를 상기 왼쪽 입력 버퍼 레지스터에 저장되어 있던 32비트의 데이터와 배타적 논리합 연산을 수행하여 연산 결과를 제1클럭에 의해서 왼쪽 레지스터에 저장하는 단계

를 포함하여 이루어진 암호화 방법.

【청구항 3】

제 1 항에 있어서,

상기 2단계에서 DES 연산 과정은 서로 반전된 클럭을 이용하여 두 개의 레지스터에 교대로 저장함으로써 8라운드의 DES 연산을 수행하는 것을 특징으로 하는 암호화 방법.

【청구항 4】

제 2 항에 있어서,

상기 사이퍼 함수부는,

32비트의 데이터 블럭를 입력받아 48비트의 데이터로 확장하는 제1확장치환부;

상기 제1확장치환부의 48비트의 데이터와 키 스케쥴로부터 생성된 보조키를 입력받는 제1배타적 논리합 연산부;

32비트의 데이터 블럭을 입력받아 48비트의 데이터로 확장하는 제2확장치환부;

상기 제2확장치환부의 48비트의 데이터와 키 스케쥴로부터 생성된 보조키를 입력받는 제2배타적 논리합 연산부;

상기 제1 및 제2 배타적 논리합 연산부로부터 48비트의 데이터를 입력받아 선택신호에 의해 그 중 하나를 선택하는 멀티플렉서;

상기 멀티플렉서로부터 48비트의 데이터를 입력받아 저장하고 32비트의 데이터로 대치하는 S-Box 치환부;

상기 S-Box 치환부로부터의 32비트의 데이터를 복사 치환하는 P-Box 치환부;

상기 P-Box 치환부로부터의 32비트의 데이터를 선택신호에 의해 두 개의 32비트의 데이터로 분배하는 디멀티플렉서

를 포함하여 이루어진 암호화 장치.

【청구항 5】

제 4 항에 있어서,

상기 멀티플렉서와 디멀티플렉서는 선택신호에 의해서 상기 제1클럭 주기의 전반부와 후반부에 각각 다른 32비트의 데이터를 연산하여 출력하게 하는 것을 특징으로 하는 암호화 장치.

【청구항 6】

제 4 항에 있어서,

상기 키 스케쥴은,

56비트의 키를 입력받아서 치환하는 제1치환선택부;

상기 제1치환선택부에 의해서 치환된 56비트의 블럭을 28비트의 두 블럭으로 나누어서 제1클럭에 의해서 저장하는 제1레지스터 및 제2레지스터;

상기 제1레지스터 및 제2레지스터의 28비트의 키 블럭을 각각 왼쪽으로 두자리, 세자리 또는 네자리씩 쉬프트하는 제1쉬프트부 및 제2쉬프트부; 및

상기 제1레지스터 및 제2레지스터의 28비트의 키 블럭을 입력받아 48비트의 보조키를 생성하는 제2치환선택부

를 구비하여 8라운드로 동작하는 제1유닛이 구성되고 제1유닛과 똑같은 구성을 갖고 다만 제2클럭이 입력되는 제2유닛을 더 포함하는 것을 특징으로 하는 암호화 장치.

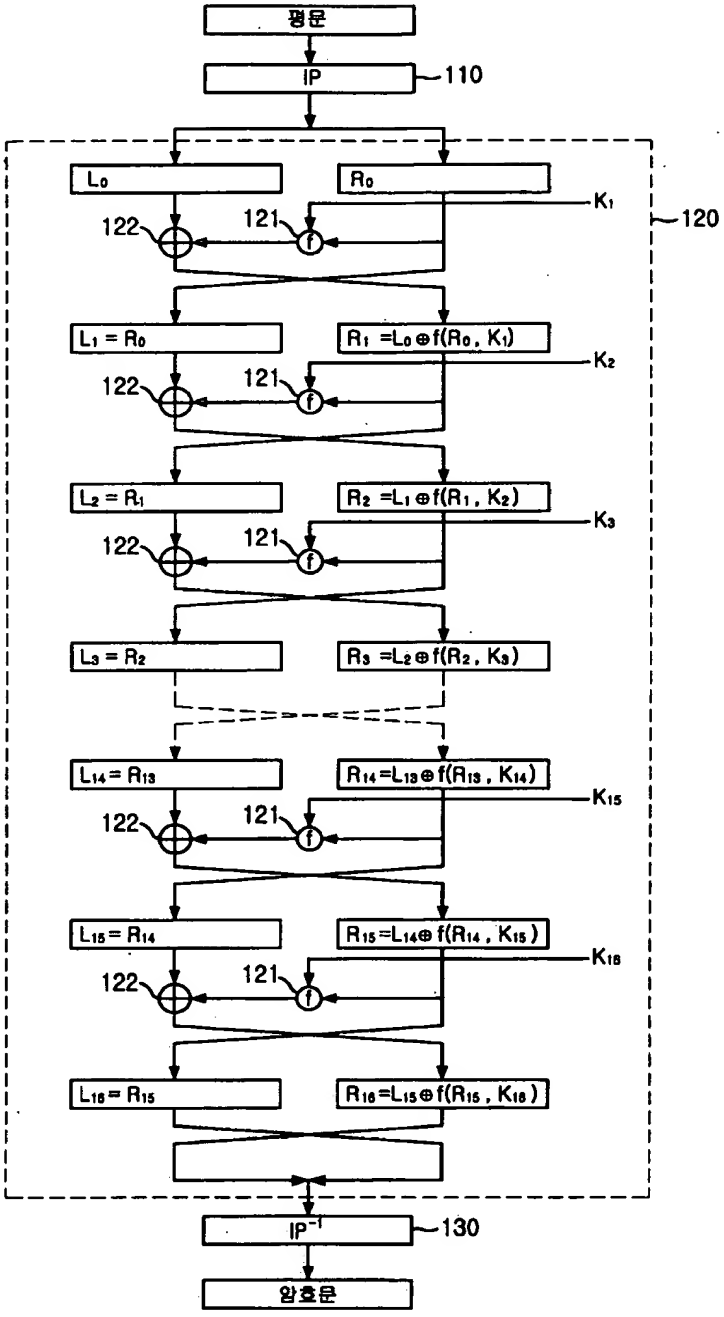
【청구항 7】

제 6 항에 있어서,

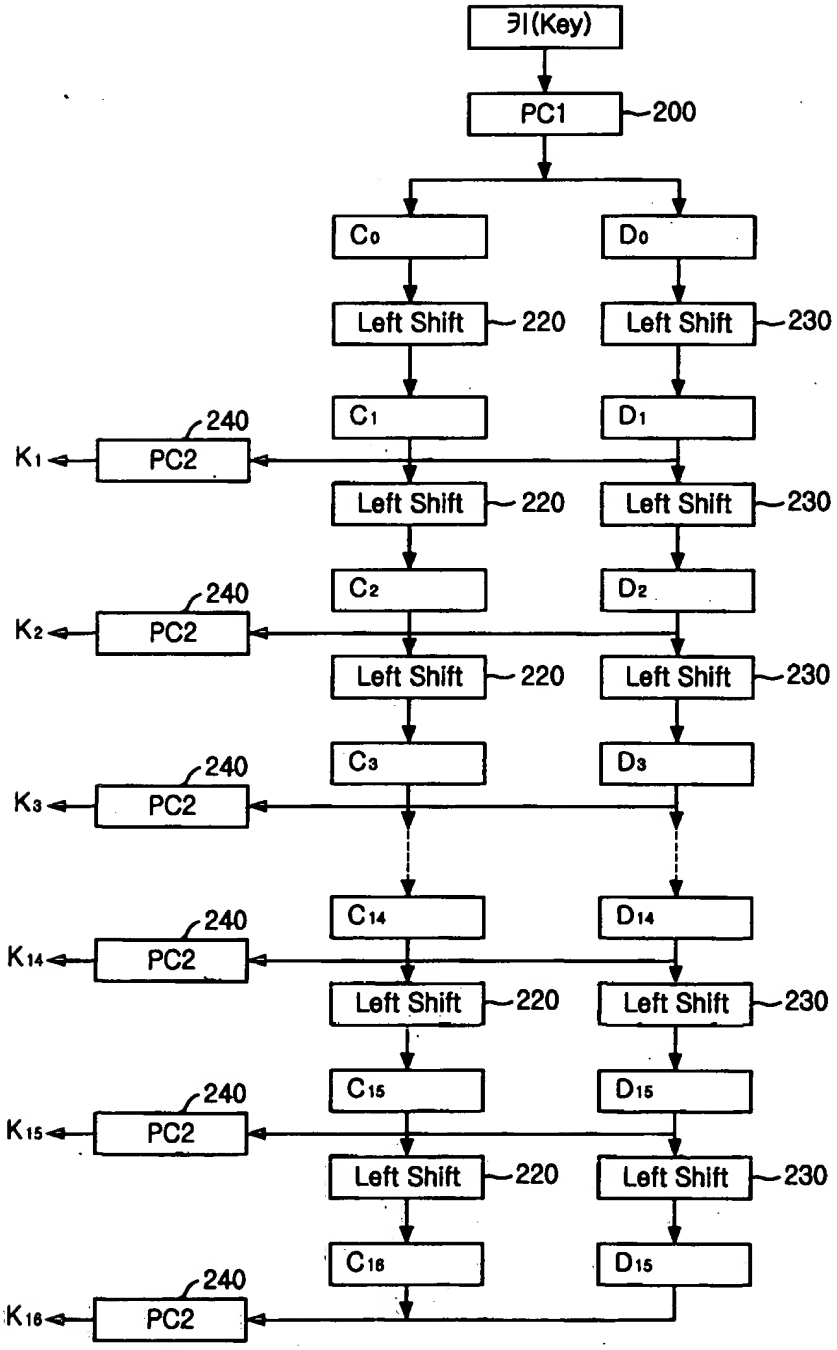
상기 제1 및 제2 치환선택부는 와이어링으로 구현 가능하고 쉬프터와 레지스터를 두 개를 사용하여 면적이 적은 것을 특징으로 하는 암호화 장치.

【도면】

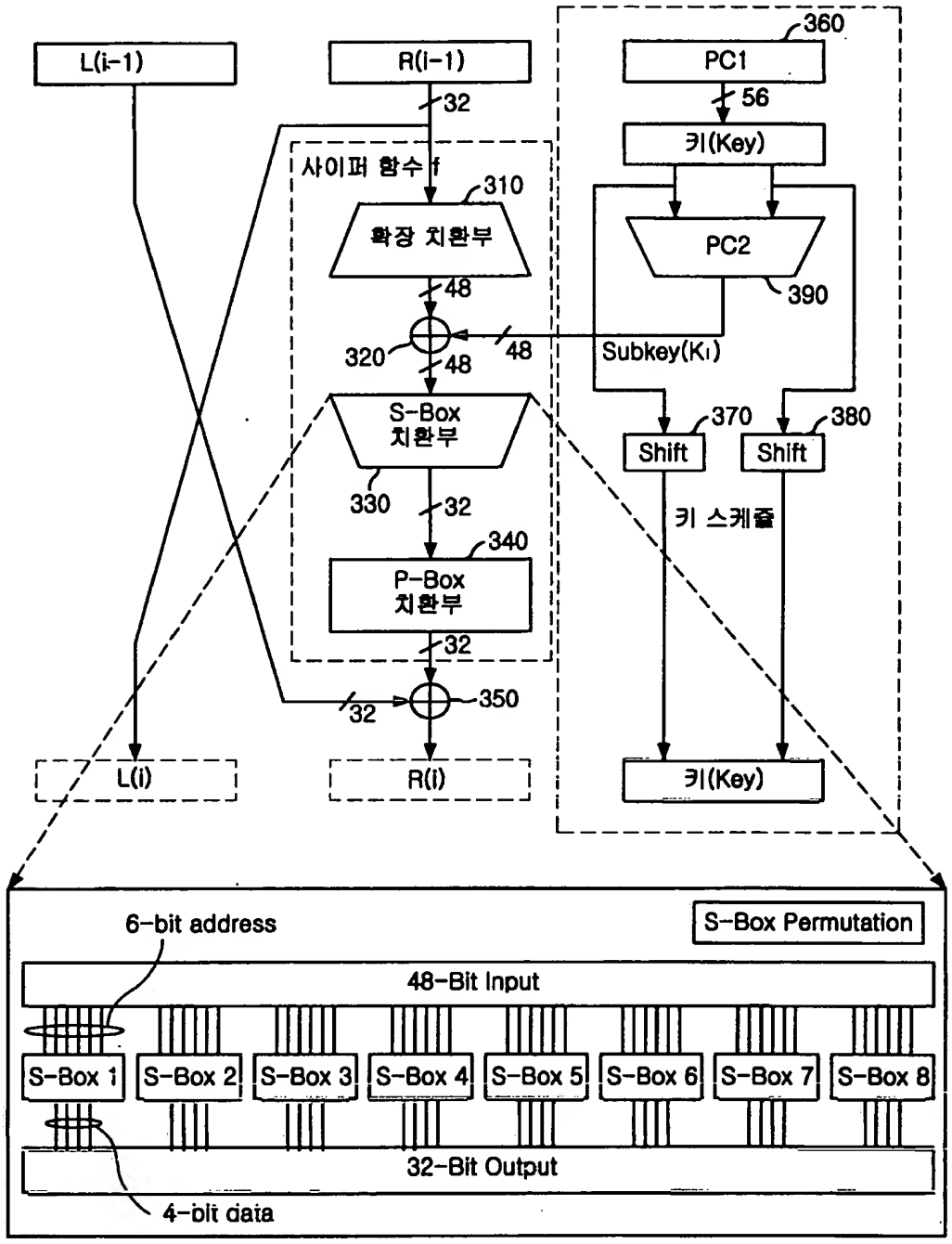
【도 1】


평문
IP
110
$L_0$
$R_0$
$K_1$
122
121
f
120
$L_1 = R_0$
$R_1 = L_0 \oplus f(R_0, K_1)$
$K_2$
$L_2 = R_1$
$R_2 = L_1 \oplus f(R_1, K_2)$
$K_3$
$L_3 = R_2$
$R_3 = L_2 \oplus f(R_2, K_3)$
$L_{14} = R_{13}$
$R_{14} = L_{13} \oplus f(R_{13}, K_{14})$
$K_{15}$
$L_{15} = R_{14}$
$R_{15} = L_{14} \oplus f(R_{14}, K_{15})$
$K_{16}$
$L_{16} = R_{15}$
$R_{16} = L_{15} \oplus f(R_{15}, K_{16})$
$IP^{-1}$
130
암호문

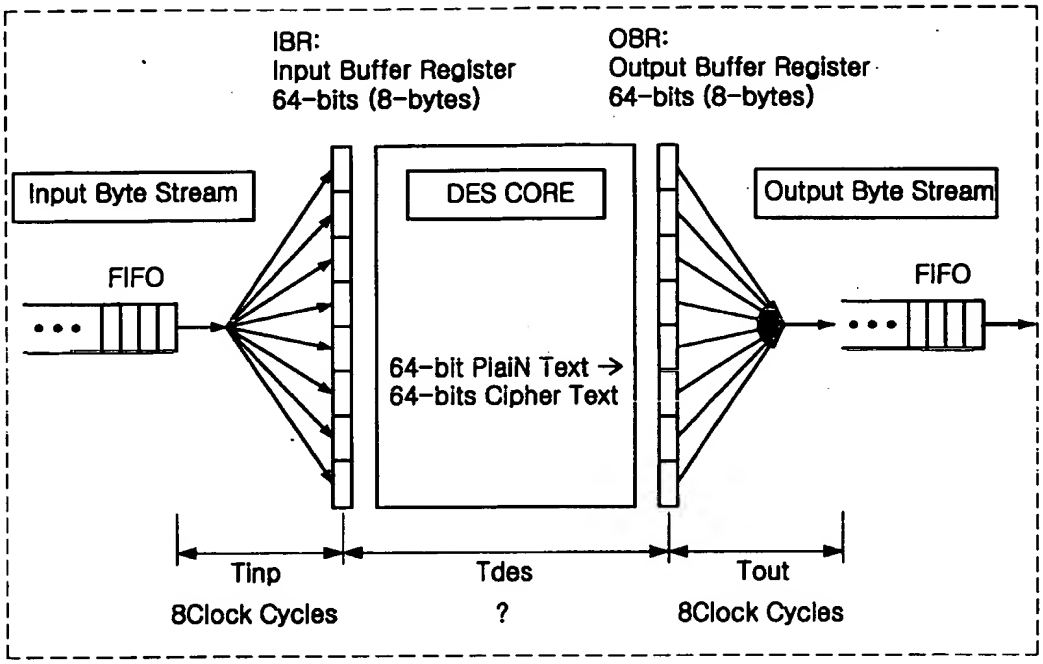
【도 2】

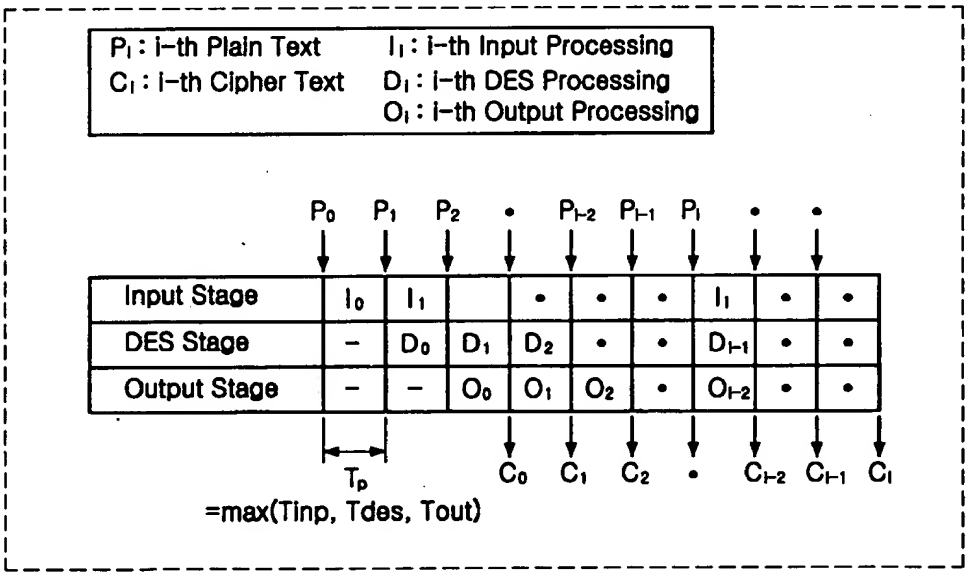

키(Key)
PC1
200
C0
D0
Left Shift
220
Left Shift
230
C1
D1
240
K1
PC2
Left Shift
220
Left Shift
230
C2
D2
240
K2
PC2
Left Shift
220
Left Shift
230
C3
D3
240
K3
PC2
C14
D14
240
K14
PC2
Left Shift
220
Left Shift
230
C15
D15
240
K15
PC2
Left Shift
220
Left Shift
230
C16
D15
240
K16
PC2

【도 3】

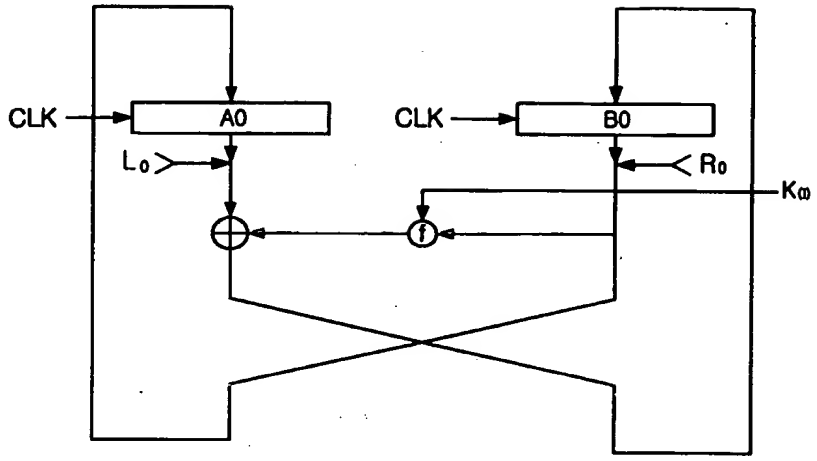

L(i-1)
R(i-1)
32
사이퍼 함수 f
310
확장 치환부
48
320
48
48
S-Box
치환부
330
32
340
P-Box
치환부
32
32
350
L(i)
R(i)
360
PC1
56
키(Key)
PC2
390
Subkey(Ki)
370
380
Shift
Shift
키 스케줄
키(Key)
6-bit address
S-Box Permutation
48-Bit Input
S-Box 1
S-Box 2
S-Box 3
S-Box 4
S-Box 5
S-Box 6
S-Box 7
S-Box 8
32-Bit Output
4-bit data

【도 4】

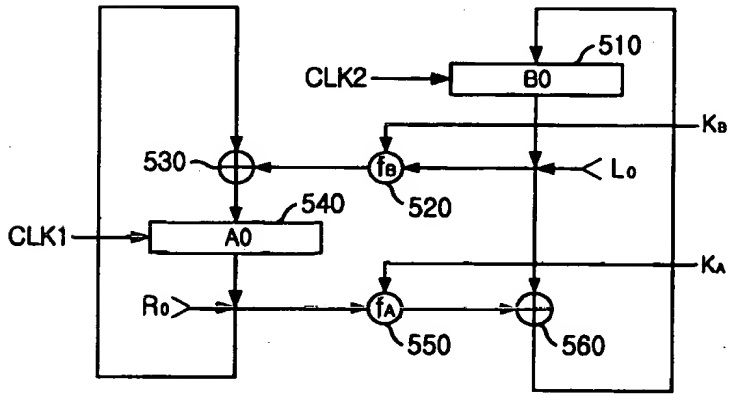

IBR:
Input Buffer Register
64-bits (8-bytes)
OBR:
Output Buffer Register
64-bits (8-bytes)
Input Byte Stream
DES CORE
Output Byte Stream
FIFO
FIFO
64-bit PlaiN Text →
64-bits Cipher Text
Tinp
8Clock Cycles
Tdes
?
Tout
8Clock Cycles

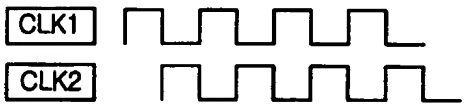

Pi : i-th Plain Text
Ci : i-th Cipher Text
Ii : i-th Input Processing
Di : i-th DES Processing
Oi : i-th Output Processing
P0 P1 P2 • Pi-2 Pi-1 Pi • •
Input Stage I0 I1 • • • Ii • •
DES Stage – D0 D1 D2 • • Di-1 • •
Output Stage – – O0 O1 O2 • Oi-2 • •
C0 C1 C2 • Ci-2 Ci-1 Ci
Tp
=max(Tinp, Tdes, Tout)

【도 5a】

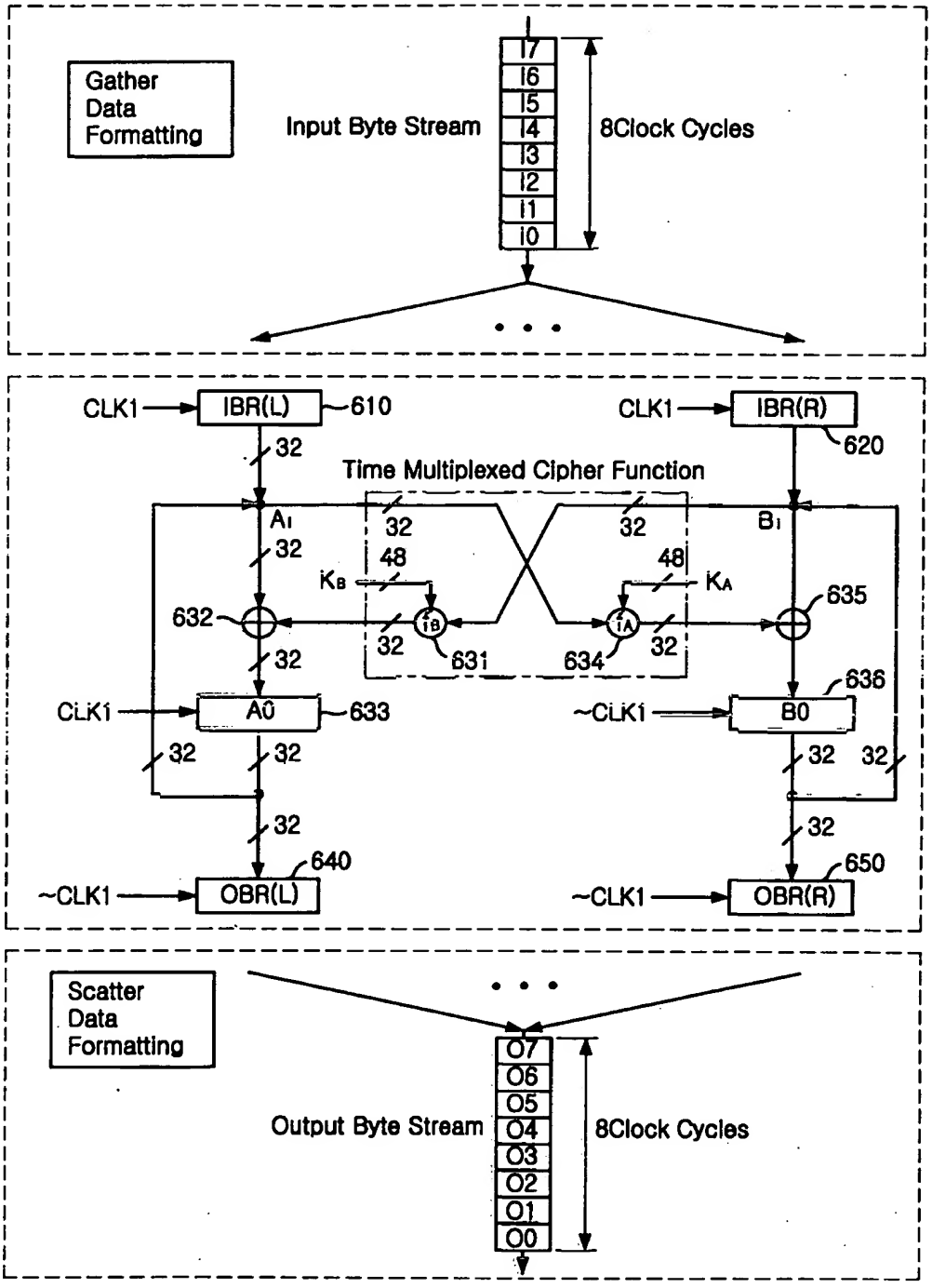

CLK
A0
CLK
B0
Lo
Ro
Kω
f

【도 5b】

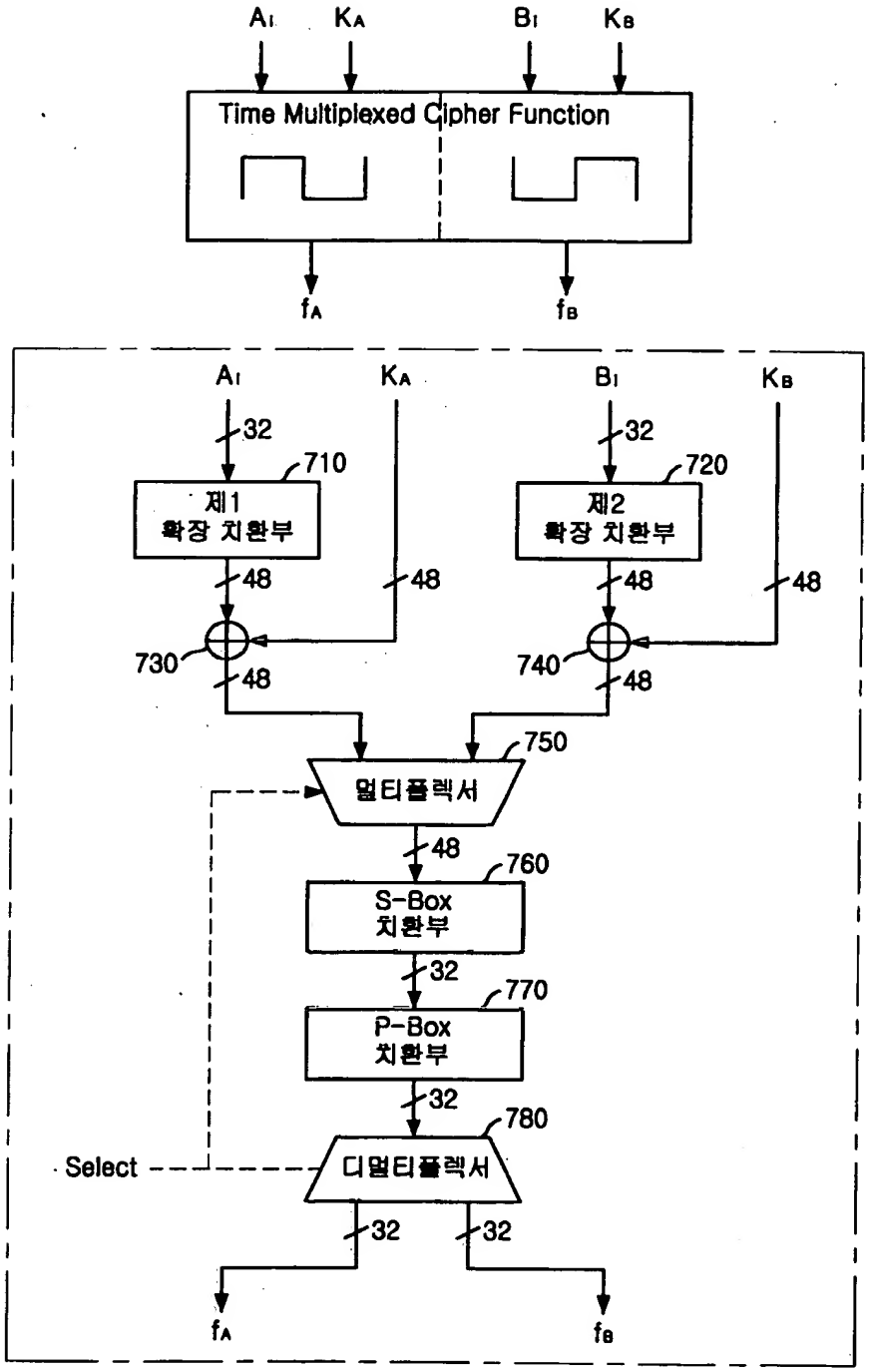

510
CLK2
B0
KB
530
fB
Lo
520
540
CLK1
A0
KA
Ro
fA
550
560


CLK1
CLK2

【도 6】


Gather Data Formatting
Input Byte Stream
I7
I6
I5
I4
I3
I2
I1
I0
8Clock Cycles
CLK1
IBR(L)
610
IBR(R)
620
32
Time Multiplexed Cipher Function
AI
BI
48
KB
KA
632
635
631
634
636
A0
633
B0
640
650
OBR(L)
OBR(R)
~CLK1
Scatter Data Formatting
Output Byte Stream
O7
O6
O5
O4
O3
O2
O1
O0

【도 7】

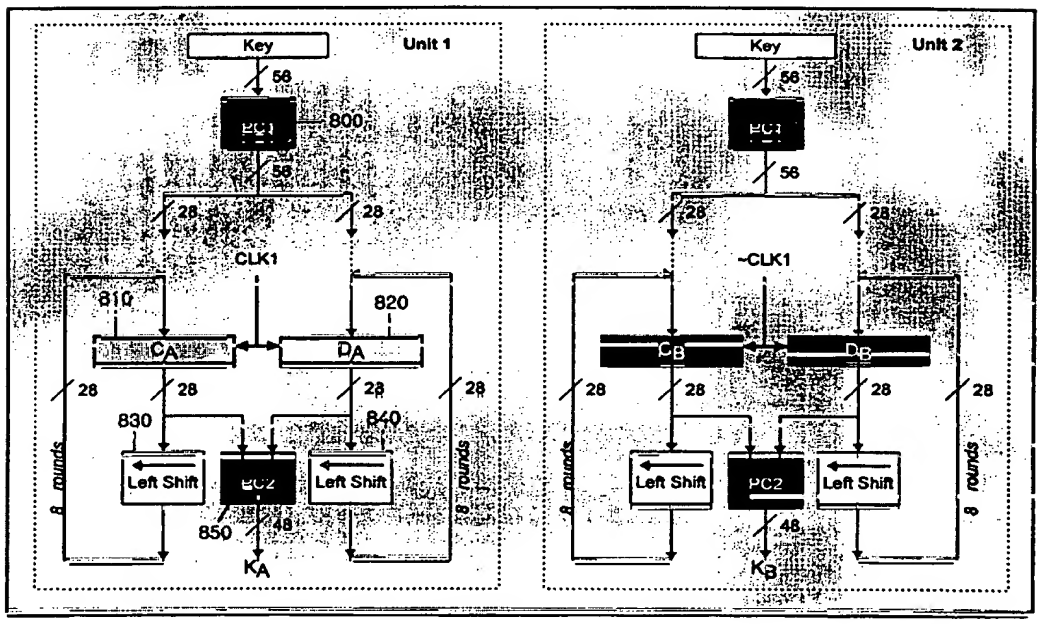

AI
KA
BI
KB
Time Multiplexed Cipher Function
fA
fB
32
710
제1
확장 치환부
720
제2
확장 치환부
48
730
740
750
멀티플렉서
760
S-Box
치환부
770
P-Box
치환부
780
디멀티플렉서
Select

【도 8】

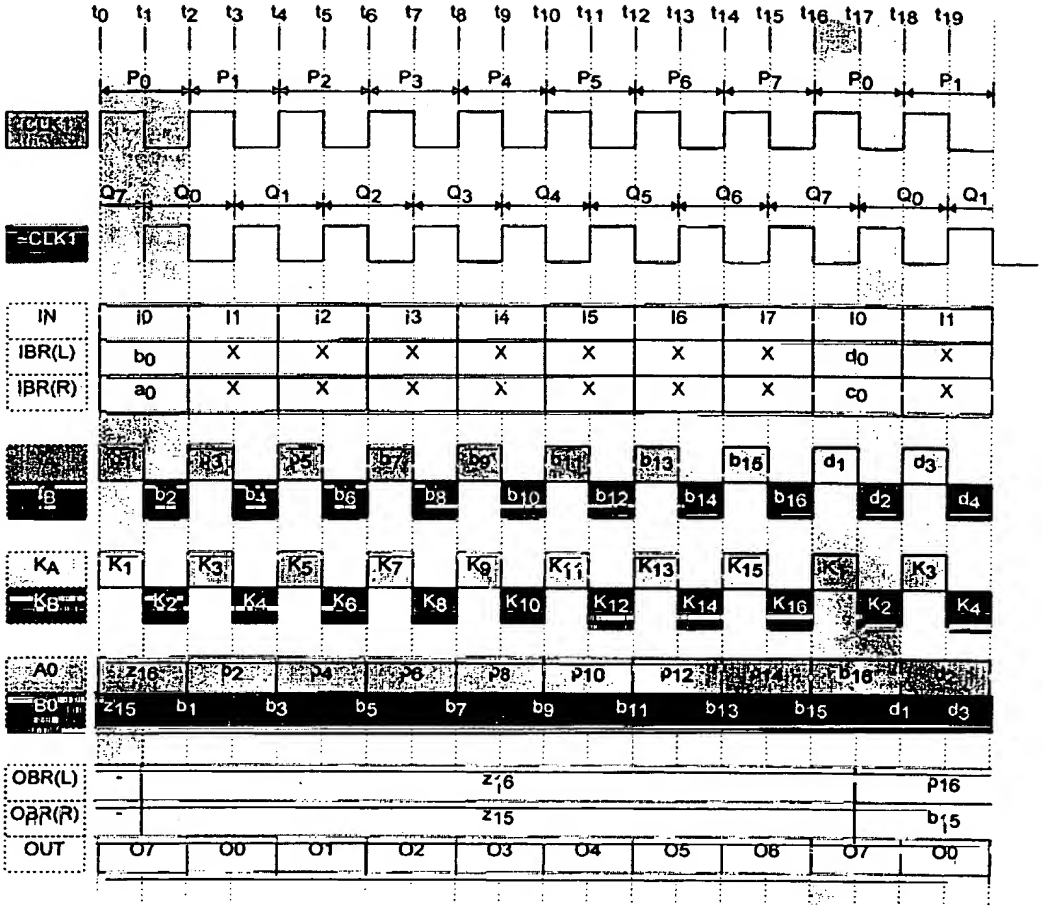
| Round | Left Shift Amount | Total Shift Amount |
|---|---|---|
| 1($P_0$) | 3 | 1 |
| 2($P_1$) | 4 | 4 |
| 3($P_2$) | 4 | 8 |
| 4($P_3$) | 3 | 12 |
| 5($P_4$) | 4 | 15 |
| 6($P_5$) | 4 | 19 |
| 7($P_6$) | 4 | 23 |
| 8($P_7$) | 2* | 27 |

* When Key is loaded first time, it should be 1.

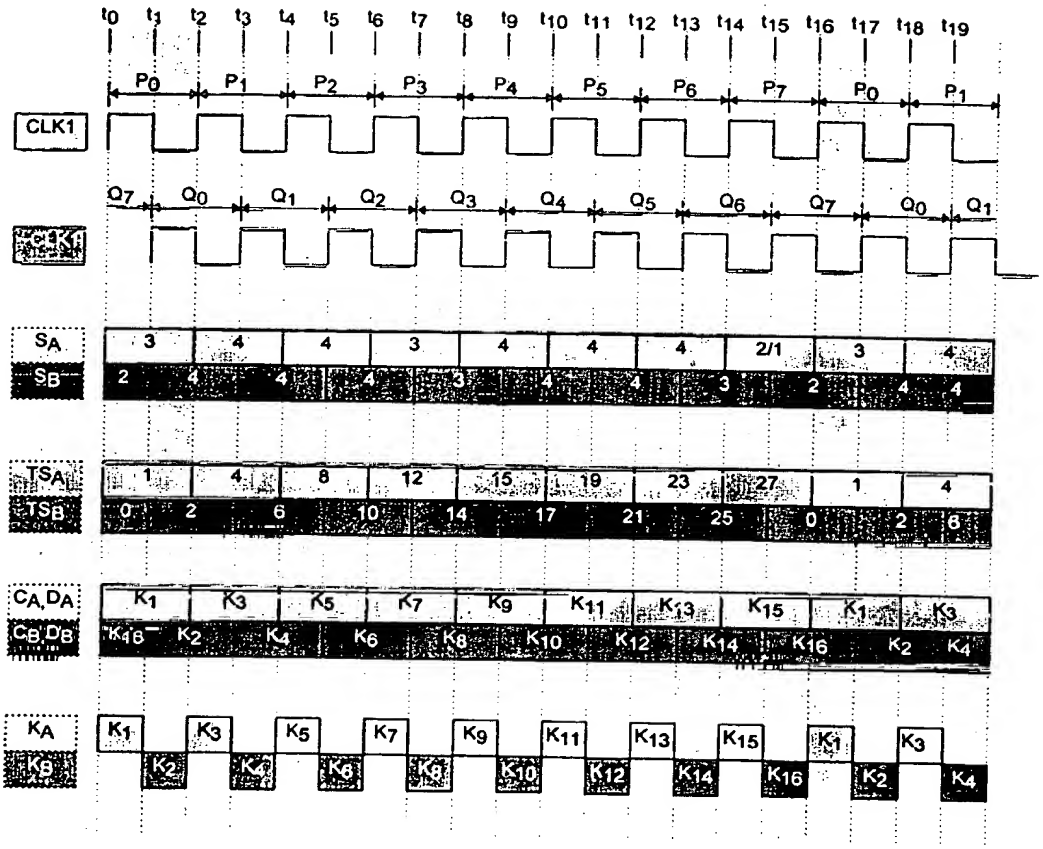
| Round | Left Shift Amount | Total Shift Amount |
|---|---|---|
| 1($Q_0$) | 4 | 2 |
| 2($Q_1$) | 4 | 6 |
| 3($Q_2$) | 4 | 10 |
| 4($Q_3$) | 3 | 14 |
| 5($Q_4$) | 4 | 17 |
| 6($Q_5$) | 4 | 21 |
| 7($Q_6$) | 3 | 25 |
| 8($Q_7$) | 2 | 0 |



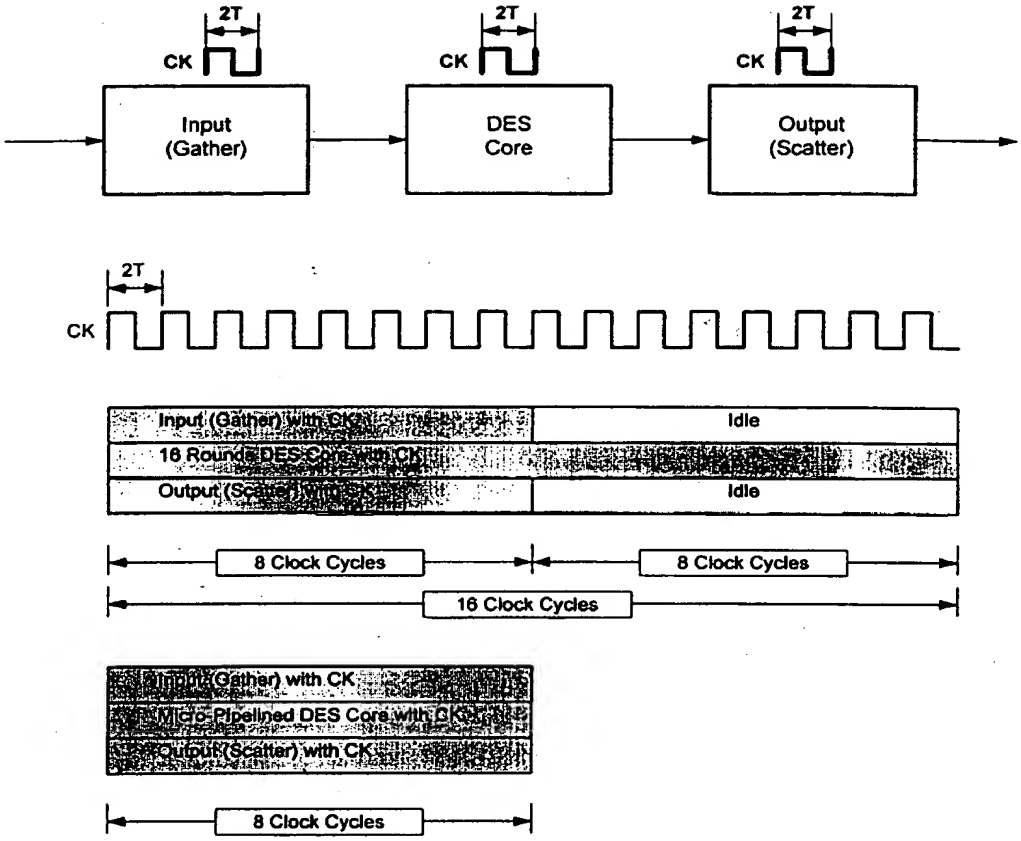
【도 9】


t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19
P0 P1 P2 P3 P4 P5 P6 P7 P0 P1
CLK1
Q7 Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q0 Q1
=CLK1
IN
IBR(L)
IBR(R)
i0 i1 i2 i3 i4 i5 i6 i7 i0 i1
b0 X X X X X X X d0 X
a0 X X X X X X X c0 X
KA
KB
A0
B0
OBR(L)
OBR(R)
OUT
O7 O0 O1 O2 O3 O4 O5 O6 O7 O0

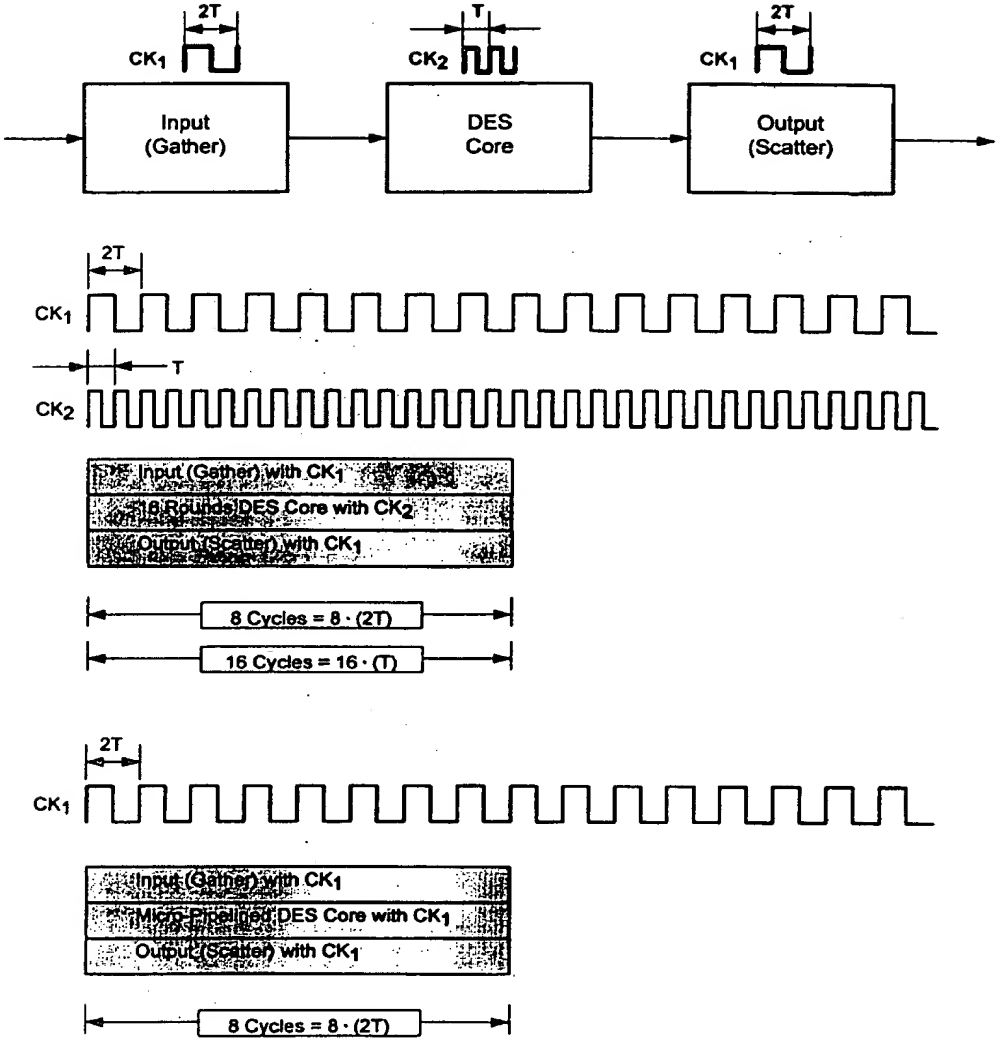
【도 10】


t0 t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19
P0 P1 P2 P3 P4 P5 P6 P7 P0 P1
CLK1
Q7 Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q0 Q1
SA
SB
TSA
TSB
CA,DA
CB,DB
KA
KB

【도 11】


2T
CK
Input
(Gather)
DES
Core
Output
(Scatter)
Input (Gather) with CK
Idle
16 Rounds DES Core with CK
Output (Scatter) with CK
8 Clock Cycles
16 Clock Cycles
Micro-Pipelined DES Core with CK

【도 12】


2T
CK1
T
CK2
2T
CK1
Input (Gather)
DES Core
Output (Scatter)
2T
CK1
T
CK2
Input (Gather) with CK1
16 Rounds DES Core with CK2
Output (Scatter) with CK1
8 Cycles = 8 · (2T)
16 Cycles = 16 · (T)
2T
CK1
Input (Gather) with CK1
Micro-Pipelined DES Core with CK1
Output (Scatter) with CK1
8 Cycles = 8 · (2T)